Regularization of Inverse Problems: Deep Equilibrium Models versus Bilevel Learning

Danilo Riccio[1] Matthias J. Ehrhardt[2] Martin Benning[3]

[1]Queen Mary University of London, United Kingdom
[2]University of Bath, United Kingdom
[3]University College London, United Kingdom

Abstract

Variational regularization methods are commonly used to approximate solutions of inverse problems. In recent years, model-based variational regularization methods have often been replaced with data-driven ones such as the fields-of-expert model [1]. Training the parameters of such data-driven methods can be formulated as a bilevel optimization problem. In this paper, we compare the framework of bilevel learning for the training of data-driven variational regularization models with the novel framework of deep equilibrium models [2] that has recently been introduced in the context of inverse problems [3]. We show that computing the lower-level optimization problem within the bilevel formulation with a fixed point iteration is a special case of the deep equilibrium framework. We compare both approaches computationally, with a variety of numerical examples for the inverse problems of denoising, inpainting and deconvolution.

1 Introduction

In inverse problems, a desired quantity can only be recovered indirectly from measurements through the inversion of a so-called forward operator. Inverse problems are found in many diverse research areas like medical and industrial imaging, signal and image processing, computer vision, or geophysics, with various applications utilizing inverse problems such as Computerized Tomography, Magnetic Resonance Imaging, or Transmission Electron Tomography, see e.g., [4, 5] and references therein. For the majority of relevant inverse problems, an inverse of the forward operator is discontinuous and usually not unique. If we focus on inverse problems with linear forward operators in finite dimensions, an inverse (if it exists) is no longer discontinuous, but can still be very sensitive to small variations in the argument if the condition number of the linear forward operator (which is then a matrix) is large.

Regularizations are parameterized operators that approximate inverses of forward operators in a continuous fashion [6, 7, 8]. Variational regularizations [5] are a special class of regularizations and usually assign the solution of a convex optimization problem to the operator input. Regularizations are equipped with hyper-parameters that are usually referred to as regularization parameters, and these regularization parameters have to be chosen dependent on the data error to guarantee that a regularization convergences to a (generalized) inverse when the error goes to zero.

A heuristic regularization parameter choice strategy that has been popularized in the past decade is the identification of optimal regularization parameters in a data-driven way with

bilevel optimization [9, 10, 11, 12, 13, 14]. In a bilevel optimization problem, an upper-level objective that is constrained by a lower-level objective is optimized. In the context of variational regularization parameter estimation, one can for example, minimize an empirical risk between the regularization operator outputs and a set of desired outputs as the upper-level optimization problem, subject to the lower-level optimization problem that the regularization operator outputs have to solve the optimization problem associated with the variational regularization.

In a development parallel to the use of bilevel optimization for the estimation of regularization parameters, more and more inverse problem solutions have been approximated with operators stemming from deep neural networks, with great empirical success [15, 16, 17, 18, 19, 20, 21, 22]. A recent development is the application of so-called deep equilibrium models [2] in the context of inverse problems [3, 23]. Deep equilibrium models are deep learning frameworks where a neural network is trained such that its fixed point approximates some ground truth solution.

None of the aforementioned papers have investigated the obvious link between deep equilibrium models and bilevel optimization. Our contributions are the framing of bilevel optimization problems with strongly convex lower-level problem as deep equilibrium models, and an empirical comparison of deep equilibrium models and bilevel optimization for three inverse problems.

The paper is organized as follows. In Section 2, we give a brief overview of the concepts of inverse problems, regularizations, bilevel optimization and deep equilibrium models. In Section 3, we choose different bilevel and deep equilibrium models for comparison and explain their architectures and other design choices. In Section 3.3, we present the three inverse problems denoising, inpainting and deblurring that we use for the numerical deep equilibrium vs bilevel optimization comparison in Section 4. We then wrap up with conclusions and give an outlook for open research questions in Section 5.

2 Deep equilibrium models and bilevel learning

In this section, we briefly recall the fundamentals of inverse problems and variational regularization. Subsequently, we describe the concepts of bilevel optimization and deep equilibrium models in this context and highlight how they are linked mathematically. We conclude this section with a comment on why learning fixed points in a naïve way does not work.

2.1 Inverse problems

Many practical applications, such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), require the reconstruction of quantities from indirect measurements. This task can be modeled as an inverse problem

$$Ku = f, \tag{1}$$

where the goal is to retrieve an unknown quantity u given the operator K and the *ideal* measured data f, with $u \in \mathbb{R}^n, f \in \mathbb{R}^m, K \in \mathbb{R}^{m \times n}$. In this paper, we limit our analysis to linear and finite-dimensional operators K, so we can think of K as a $m \times n$ matrix.

If the operator K is invertible, the unique solution of (1) is $u = K^{-1}f$. In case the operator K is not invertible, the Moore–Penrose inverse K^+ can be used to approximate the inverse

instead. The Moore–Penrose inverse is the minimum norm solution of the normal equation $K^\top Ku = K^\top f$, which means that it minimizes the least squares function $J(u) = \frac{1}{2}\|Ku - f\|^2$, where $\|\cdot\|$ is the Euclidean norm, with minimal Euclidean norm $\|u\|$. By using the Moore–Penrose inverse to solve (1) for u, we solve

$$\bar{u} = K^+ f. \tag{2}$$

In most applications, this problem is usually ill-conditioned. In other words, as soon as we consider the presence of noise affecting the measurement, the worst-case error between retrieved data and desired data is strongly amplified. In this paper we will consider any measurement error to be additive, i.e.,

$$Ku + \delta = f^\delta, \tag{3}$$

$\delta, f^\delta \in \mathbb{R}^m$, where subscript δ is used to remind that the measured data f^δ is affected by (additive) noise.

To avoid getting an ill-conditioned matrix like in (2), a standard procedure known as *variational regularization* is to define a new cost function J_R by adding a regularizer $\mathcal{R}$ to the cost function $\mathcal{D}$, i.e.,

$$J_{\mathcal{R}}(u) = \lambda \mathcal{D}(Ku, f^\delta) + \mathcal{R}(u)\,, \tag{4}$$

where $\mathcal{D}$ is a proper, convex and continuously differentiable function in the first argument, $\mathcal{R}$ is a proper, convex and lower semi-continuous function, and $\lambda > 0$ is the so-called regularization parameter that balances the influence of the data term J and the regularizer $\mathcal{R}$. One of the most famous examples in the literature is the Tikhonov regularizer $\mathcal{R}(u) = \frac{1}{2}\|u\|^2$ (cf. [24]). For example, Tikhonov regularization with the least squares cost function in (4) leads to

$$u_\lambda = (\lambda K^\top K + I_n)^{-1} K^\top f^\delta, \tag{5}$$

where $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix.

More recently, model-based regularizers such as Tikhonov-type regularization have been replaced by data-driven regularizers of the form $\mathcal{R}_\Psi(x)$, where the subscript Ψ indicates that $\mathcal{R}_\Psi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a parameterized function with parameters $\Psi \in \mathbb{R}^p$. Examples for such data-driven variational regularizations are Markov Random Field priors like the fields-of-expert regularizer [1], which in combination with (4) was popularized for approximating inverse problems solutions in [25]:

$$u_{\lambda,\Psi} \in \operatorname*{arg\,min}_{u\in\mathbb{R}^n} \left\{ \frac{\lambda}{2}\|Ku - f^\delta\|^2 + \sum_{i=1}^{r} \phi_i(A_i u) \right\}. \tag{6}$$

Here $\{\phi_i\}_{i=1}^r$ are proper, convex and lower semi-continuous functions and the parameters are $\Psi = \{A_i\}_{i=1}^r$. Optimal parameters Ψ can be found by training the model for given pairs (u, f^δ) with different techniques. In this paper we compare two of them, namely *bilevel learning* and *deep equilibrium models*.

2.2 Deep equilibrium models

Deep equilibrium models [2] are deep neural networks that have fixed points, and those fixed points are trained to match data samples from a training data set. Suppose we denote by

$G(u, \Psi) : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ our deep neural network with parameters $\Psi \in \mathbb{R}^p$, then training a deep equilibrium model can be formulated as the constrained optimization problem

$$\min_{\Psi} J(u) \qquad \text{subject to} \qquad u = G(u, \Psi)\,, \tag{7}$$

where J is the data fidelity term between the argument u and the corresponding ground truth $u^\dagger$ (e.g., mean squared error, or cross entropy function). We now want to re-formulate (7) with the help of a Lagrange multiplier μ to the saddle-point problem $\min_{u,\Psi} \max_{\mu} \mathcal{L}(u, \Psi, \mu)$ with

$$\mathcal{L}(u, \Psi, \mu) = J(u) + \langle \mu, u - G(u, \Psi) \rangle\ , \tag{8}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Computing the optimality system of (8), i.e., computing the partial derivatives of $\mathcal{L}$ with respect to the individual arguments and setting them to zero, yields the nonlinear system of equations

$$u^* = G(u^*, \Psi^*)\,, \tag{9a}$$

$$0 = \nabla J(u^*) + \left(I - (\partial_{u^*} G(u^*, \Psi^*))^\top\right) \mu^*\,, \tag{9b}$$

$$0 = \partial_{\Psi^*} G(u^*, \Psi^*)^\top \mu^*\,, \tag{9c}$$

where $\partial_{u^*} G(u^*, \Psi^*)$ and $\partial_{\Psi^*} G(u^*, \Psi^*)$ denote the Jacobian matrices of G with respect to u^* and Ψ^*, respectively. Note that in order to compute a solution u^* of (9a) we are required to solve a fixed point problem and further require that such a fixed point exists. In order to compute a solution μ^* to (9b), we need to solve the linear system

$$\mu^* = -\left(I - (\partial_{u^*} G(u^*, \Psi^*))^\top\right)^{-1} \nabla J(u^*).$$

In [2] it was proposed to also formulate this problem as a fixed point problem, i.e., we aim to find μ^* that satisfies

$$\mu^* = (\partial_{u^*} G(u^*, \Psi^*))^\top \mu^* - \nabla J(u^*)\,.$$

It is important to emphasize that, as for the fixed point u^*, we obviously require existence of μ^* in order to be able to compute it. Assuming that we can compute both u^* and μ^*, we can then compute the gradient of $\mathcal{L}$ with respect to the network parameters Ψ^*, for instance with a gradient-based iterative algorithm like gradient descent,

$$\Psi^*_{j+1} = \Psi^*_j - \tau\, \partial_{\Psi^*} G(u^*_j, \Psi^*_j)^\top \mu^*_j\,,$$

where $\{\Psi^*_j\}_{j=1}^\infty$, $\{u^*_j\}_{j=1}^\infty$ and $\{\mu^*_j\}_{j=1}^\infty$ are sequences to approximate Ψ^*, u^* and μ^*, and where every u^* satisfies $u^*_j = G(u^*_j, \Psi^*_j)$, while every μ^*_j satisfies $\mu^*_j = (\partial_{u^*} G(u^*_j, \Psi^*_j))^\top \mu^*_j - \nabla J(u^*_j)$. Here, the parameter τ is a positive step-size parameter that controls the length of the step in the direction of the negative gradient. In practice, any first-order optimization method other than gradient descent can also be used to find optimal parameters Ψ^*.

In the context of inverse problems of the form (1), we can construct many meaningful deep equilibrium models. In [3], one of the methods of consideration is a so-called DeProx-type

method, where a neural network is composed with a gradient descent step on the mean-squared error of the forward model output and the measurement data (which in this context is also known as a step of Landweber regularization), i.e.,

$$u^{k+1} = \mathcal{N}_\Psi\left(u^k - \tau\lambda\, K^\top\left(Ku^k - f^\delta\right)\right), \tag{10}$$

for a neural network $\mathcal{N}$ with parameters Ψ. Assuming the neural network is 1-Lipschitz, i.e.,

$$\|\mathcal{N}_\Psi(u) - \mathcal{N}_\Psi(v)\| \leq \|u - v\|,$$

for all $u, v \in \mathbb{R}^n$ and choosing τ such that $\tau < 2/(\lambda\|K\|^2)$ is satisfied, then we observe that u^* with

$$u^* = \mathcal{N}_\Psi\left(u^* - \tau\lambda K^\top\left(Ku^* - f^\delta\right)\right) \tag{11}$$

is a fixed point of iteration (10), which we can conclude from Banach's fixed point theorem if $K^\top K$ has full rank (in which case the fixed point is also unique); if $K^\top K$ does not have full rank, the mapping is only nonexpansive and non-emptiness of the fixed point set has to be verified (cf. [26]). Hence, we can guarantee convergence of the sequence. The Lipschitz continuity of the mapping $\mathcal{N}_\Psi\left(u - \tau\lambda K^\top\left(Ku - f^\delta\right)\right)$ implies continuity, so we can characterize the fixed point via

$$\begin{aligned} u^* &= \lim_{k\to\infty} u^k = \lim_{k\to\infty} \mathcal{N}_\Psi\left(u^{k-1} - \tau\lambda K^\top\left(Ku^{k-1} - f^\delta\right)\right) \\ &= \mathcal{N}_\Psi\left(\lim_{k\to\infty}\left(I - \tau\lambda K^\top K\right)u^{k-1} + \tau\lambda K^\top f^\delta\right) = \mathcal{N}_\Psi\left(\left(I - \tau\lambda K^\top K\right)u^* + \tau\lambda K^\top f^\delta\right). \end{aligned}$$

This means that for the operator $G(u, \Psi) := \mathcal{N}_\Psi\left(u - \tau\lambda K^\top\left(Ku - f^\delta\right)\right)$, the deep equilibrium problem (7) is well-defined if the conditions outlined above are met.

Another method that was considered in [3] is the so-called DeGrad-type method, which is motivated by gradient descent where a neural network is used to replace the gradient of a regularization term, i.e.,

$$u^{k+1} = u^k - \tau\left(\lambda\, K^\top\left(Ku^k - f^\delta\right) + \mathcal{N}_\Psi(u^k)\right). \tag{12}$$

Making similar assumptions as in the DeProx case, we can assume that a (unique) fixed point exists, so that the deep equilibrium problem (7) is well-defined again.

2.3 Bilevel learning

Following the description of data-driven variational regularization models such as (6) in Section 2.1, we want to briefly recall how the parameters Ψ of the regularizer can be trained with the help of bilevel learning. In this context, we can formulate the bilevel optimization problem as

$$\min_\Psi J(u^*), \tag{13a}$$

subject to

$$u^* \in \arg\min_u\left\{\frac{\lambda}{2}\|Ku - f^\delta\|^2 + \mathcal{R}_\Psi(u)\right\}, \tag{13b}$$

where (13a) and (13b) are the upper-level and the lower-level problems, respectively. Here, J denotes a convex and continuously differentiable loss function, and $\mathcal{R}_\Psi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper, convex and lower semi-continuous function that is parameterized by parameters Ψ. Note that due to the convexity of the lower-level problem, u^* will always be a global minimizer. General existence results of (13) can for example be found in [27]. Traditionally, (13) is solved with the help of gradient-based optimization methods. Historically, in order to compute the gradient of the upper-level optimization problem with regards to Ψ it is common to make use of the implicit function theorem (cf. [28, Theorem 4.E] and [29, Corollary 4.34]). If we assume that $\mathcal{R}_\Psi$ is differentiable with respect to its argument, we can characterize the solution of (13b) via the optimality condition

$$0 = S(u, \Psi) = \lambda K^\top \left(Ku - f^\delta \right) + \nabla \mathcal{R}_\Psi(u) \,.$$

If we further assume that S is strictly differentiable and that $\nabla_u S(u, \Psi)$ is invertible, we can compute $\nabla J(u^*(\Psi))$ via the implicit function theorem, i.e.

$$\nabla J(u^*(\Psi)) = \left(\nabla_u S(u, \Psi)\right)^{-1} \nabla_\Psi S(u, \Psi) \,.$$

Please note that for S to be strictly differentiable, we require $\mathcal{R}_\Psi$ to be twice differentiable with regards to its argument. The invertibility of $\mathcal{R}_\Psi$ can usually be achieved if we make $\mathcal{R}_\Psi$ strictly convex, for example with an additional elliptic regularization term as in [9].

In the following section, we want to highlight that bilevel optimization can be viewed as a special case of the deep equilbirium model if we replace the lower-level optimization problem with an equivalent fixed-point problem.

2.4 Bilevel learning as a deep equilibrium model

Under suitable conditions on $\mathcal{R}_\Psi$ (for example strong convexity), we can conclude uniqueness of u^* in (13b). If the solution u^* is unique, its computation can also be replaced by a suitable fixed point iteration. To give an example, if $\mathcal{R}_\Psi$ is considered L-smooth, i.e. $\mathcal{R}_\Psi$ is continuously differentiable and $\nabla \mathcal{R}_\Psi$ is Lipschitz-continuous with Lipschitz constant L, one can minimize the lower-level problem (13b) via an iterative strategy such as gradient descent, i.e.,

$$u^{k+1} = u^k - \tau \left(\lambda K^\top (Ku^k - f^\delta) + \nabla \mathcal{R}_\Psi(u^k) \right) , \tag{14}$$

for a sequence $\{u^k\}_{k=0}^\infty$ with arbitrary initial value u^0 and a positive step-size parameter τ. If τ is chosen appropriately, then convergence of (14) to u^* can be guaranteed. As a consequence, we can replace (13b) with the fixed point constraint

$$u^* = u^* - \tau \left(\lambda K^\top (Ku^* - f^\delta) + \nabla \mathcal{R}_\Psi(u^*) \right) \tag{15}$$

and therefore reformulate (13) to a saddle-point problem of the form of (8) with $G(u^*, \Psi) = u^* - \tau \left(\lambda K^\top (Ku^* - f^\delta) + \nabla \mathcal{R}_\Psi(u^*) \right)$. Note that, depending on the choice of $\mathcal{R}_\Psi$, instead of gradient descent many other optimization algorithms can be chosen, which leads to a variety of different fixed point equations $u^* = G(u^*, \Psi)$ with the same fixed point u^* that can be chosen in (8).

We want to emphasize that every minimization problem $\min_u F(u)$ with proper, convex and lower semi-continuous F can naïvely be turned into a fixed-point iteration of the form

$$u^{k+1} = \arg\min_u \left\{ F(u) + \frac{1}{2\tau}\|u - u^k\|^2 \right\}, \tag{16}$$

for a positive step-size parameter τ. Minimizing (16) is obviously as difficult as minimizing F itself, so other fixed-point iterations such as (15) are usually more suitable in practice if applicable. However, if F has a minimum, then u^k as defined in (16) converges to the set of minimizers of F and $F(u^k)$ converges to its optimal value [30]. This implies that bilevel learning problems with proper, convex and lower semi-continuous lower level problems form a subset of deep equilibrium methods.

Vice versa, every fixed-point problem can naïvely be converted into a (potentially non-convex) optimization problem by converting a fixed-point equation like $u^* = G(u^*)$ into $\min_u H(u - G(u))$, for a non-negative continuous function H with $H(0) = 0$. An example for H is the squared Euclidean norm, i.e. $H(v) = \|v\|^2/2$. In this setting it is obvious that $u^* = G(u^*)$ is a global minimizer of $H(u - G(u))$, assuming that the fixed-point exists. However, a more interesting question is whether the class of fixed-point operators is strictly larger than the class of fixed-point operators arising from the computational minimization of optimization problems. A thorough mathematical discussion of this question is beyond the scope of this paper, but it is clear that certain vector-fields, such as $C^\top f(Ax)$ for a function f, a vector x and two matrices A and C, cannot be characterized as gradients of functions if $C \neq A$ because the curl of the vector-fields is non-zero, which means the vector fields are not conservative.

2.5 Why naïvely learning fixed points does not work

We conclude this section by briefly addressing the problem of naïvely learning fixed points by minimizing empirical risks that measure the deviation between model output and desired output data $u^\dagger$. Suppose we choose the De-Prox architecture (10) from Section 2.2. Instead of solving (7) with G defined as $G(u, \Psi) = \mathcal{N}_\Psi\left(u - \tau\lambda K^\top\left(Ku - f^\delta\right)\right)$ for $J(u) := \frac{1}{2}\|u - u^\dagger\|^2$, we could naïvely train the parameters Ψ by minimizing L defined as

$$L(u^\dagger, \Psi) := \frac{1}{2}\left\|\mathcal{N}_\Psi\left(u^\dagger - \tau\lambda K^\top\left(Ku^\dagger - f^\delta\right)\right) - u^\dagger\right\|^2 \tag{17}$$

with respect to Ψ. Suppose $f^\delta = Ku^\dagger + \delta$, for some perturbation δ, then Problem (17) is the same as minimizing an empirical risk for a denoising autoencoder, i.e.,

$$\min_\Psi \frac{1}{2}\left\|\mathcal{N}_\Psi\left(u^\dagger + \tau\lambda K^\top\delta\right) - u^\dagger\right\|^2. \tag{18}$$

The problem with (18) is that the network $\mathcal{N}_\Psi$ is only trained to remove the term $\tau\lambda K^\top\delta$. However, if K is underdetermined (like in the inpainting application later), then $\mathcal{N}_\Psi$ in iteration (10) also has to be able to combat any lack of information that stems from the underdeterminedness of K (for initial values other than $u^\dagger$). For that reason, solving (7) is superior over (18) in this particular context.

Similar considerations can be drawn if we choose the De-Grad architecture (12). If we assume that the sequence $\{u^k\}_{k=0}^\infty$ converges to the fixed point u^*, and that the reconstruction

is perfect (i.e., $u^* = u^\dagger$), then we can train the parameters Ψ by minimizing L defined as

$$L(u^\dagger, \Psi) := \frac{1}{2} \left\| \lambda K^\top (K u^\dagger - f^\delta) + \mathcal{N}_\Psi(u^\dagger) \right\|^2 \tag{19}$$

with respect to Ψ. If we again assume $f^\delta = K u^\dagger + \delta$ for some perturbation δ, then Problem (19) is equivalent to

$$\min_\Psi \frac{1}{2} \left\| \mathcal{N}_\Psi(u^\dagger) - \lambda K^\top \delta \right\|^2 , \tag{20}$$

which means that in this case the network $\mathcal{N}_\Psi$ is trained to approximate $\lambda K^\top \delta$. Again, if K is underdetermined, then iteration (12) has to compensate for the lack of information that stems from the underdeterminedness of K, which is not the case when (20) is solved instead. In Section 4.6 we will train a model by minimizing (20) and empirically verify that the results obtained with this model are not as good as the ones we get when we minimize (7) instead.

3 Architecture design and implementation

We compare deep equilibrium and bilevel learning methods empirically for three different inverse problems. In general, it is very difficult to have a fair comparison between different architectures, as their expressivity is not only determined by the number of parameters or the choice of activation functions, but the complex interplay of all architecture design choices as well as other factors like the optimization method that is used for training the models, or the choice of loss and regularization functions, to name only a few. We have therefore decided to choose one deep equilibrium model that belongs to the class of deep equilibrium gradient descent, cf. [3, Section 3.1]. We compare this model with a bilevel approach where the lower-level problem is computed via gradient descent, and both architectures are chosen to be as similar as possible. The individual models are described in detail in the following sections.

3.1 Deep equilibrium models

In this section we describe the specific deep equilibrium models that we use for the comparison of numerical results in Section 4. For a fairer comparison with bilevel optimization methods, we focus on models of type DeGrad as described in Section 2.2.

3.1.1 Deep equilibrium gradient descent

Following [3, Section 3.1], we define the deep equilibrium gradient descent method for the approximation of inverse problems solutions as

$$u^{k+1} = u^k - \tau \left(\lambda K^\top (K u^k - f^\delta) + \mathcal{N}_\Psi(u^k) \right) , \tag{21}$$

where $\mathcal{N}_\Psi : \mathbb{R}^n \to \mathbb{R}^n$ is a neural network with parameters Ψ. For the comparison with bilevel optimization methods, we will choose $\mathcal{N}_\Psi$ as

$$\mathcal{N}_\Psi(u) = \gamma \, C^\top \sigma(Au + b) , \tag{22}$$

for parameters $\Psi = (A, C, b)$ with matrices $A, C \in \mathbb{R}^{q \times r}$ and bias vector $b \in \mathbb{R}^q$, activation function $\sigma : \mathbb{R}^q \to \mathbb{R}^q$ and positive constant γ. Please note that (22) is not necessarily

a gradient of a function with argument u, unless $C = A$ and suitable choices of σ. For consistency in the numerical examples below, we will choose the same activation functions discussed in Section 3.2.

3.2 Bilevel learning models

In order to effectively compare bilevel optimization with deep equilibrium models, we replace the solution of (13b) with the fixed point of (15). We choose $\mathcal{R}_\Psi(u)$ to be of the form

$$\mathcal{R}_\Psi(u) = \gamma \inf_v \left(\frac{1}{2}\|v - Au - b\|^2 + \mathcal{R}(v) \right) , \tag{23}$$

which is the Moreau–Yosida regularization [31, 32] of the proper, convex and lower semi-continuous function $\mathcal{R} : \mathbb{R}^s \to \mathbb{R} \cup \{\infty\}$ composed with the affine-linear transformation $A \cdot + b$. Here $\Psi = (A, b)$ denote the model parameters, which are a matrix $A \in \mathbb{R}^{s \times n}$ and a bias vector $b \in \mathbb{R}^s$, and γ is a positive parameter. The gradient of $\mathcal{R}_\Psi$ with respect to argument u reads

$$\nabla \mathcal{R}_\Psi(u) = \gamma A^\top \left(Au + b - \operatorname{prox}_{\mathcal{R}} \left(Au + b \right) \right) = \gamma A^\top \operatorname{prox}_{\mathcal{R}^*} \left(Au + b \right) , \tag{24}$$

cf. [30, Proposition 12.30], where $\operatorname{prox}_{\mathcal{R}} : \mathbb{R}^s \to \mathbb{R}^s$ denotes the proximal map [33] of $\mathcal{R}$, i.e.,

$$\operatorname{prox}_{\mathcal{R}}(w) = \operatorname*{arg\,min}_{v \in \mathbb{R}^s} \left\{ \frac{1}{2}\|v - w\|^2 + \mathcal{R}(v) \right\} ,$$

and $\operatorname{prox}_{\mathcal{R}^*}$ denotes the proximal map of the convex conjugate $\mathcal{R}^*$ of $\mathcal{R}$ (cf. [34]) that is defined as

$$\mathcal{R}^*(p) := \sup_v \left(\langle v, p \rangle - \mathcal{R}(v) \right) .$$

Note that (24) is equivalent to (22) if $C = A$ and if $\sigma = \operatorname{prox}_{\mathcal{R}^*}$ for some proper, convex and lower semi-continuous $\mathcal{R}$ with conjugate $\mathcal{R}^*$.
Suppose we define χ_C as the characteristic function over the convex set C, i.e.,

$$\chi_C(u) := \begin{cases} 0 & u \in C \\ \infty & u \notin C \end{cases} .$$

If we choose $\mathcal{R}(u) = \chi_{C_1}(u)$ with $C_1 = \{0\}$, we observe $\mathcal{R}_\Psi(u) = \frac{1}{2}\|Au + b\|^2$ and $\mathcal{R}^*(p) = 0$. The proximal map of $\mathcal{R}^*$ then simply reduces to the identity, i.e., $\operatorname{prox}_{\mathcal{R}^*}(w) = w$, and (24) reduces to $\nabla \mathcal{R}_\Psi(u) = \gamma A^\top (Au + b)$.

Another interesting example is $\mathcal{R}(u) = \chi_{C_2}(u)$ with $C_2 = (-\infty, 0]^s$, where we recover the elementwise Rectified Linear Unit (ReLU) [35] activation function $\sigma(x) = \operatorname{ReLU}(x)$ with

$$(\operatorname{ReLU}(x))_i := \begin{cases} x_i & x_i \geq 0 \\ 0 & x_i < 0 \end{cases} ,$$

for the proximal map of the conjugate. In this case, (24) reduces to $\nabla \mathcal{R}_\Psi(u) = \gamma A^\top \operatorname{ReLU}(Au + b)$.

If we choose $\mathcal{R}(v) = \varepsilon\|v\|_1 = \varepsilon \sum_{j=1}^{m} |v_j|$, we have $\mathcal{R}^*(p) = \chi_{C_3}(u)$ with $C_3 = \{u \,|\, \|u\|_\infty \leq \epsilon\}$ and

$$\operatorname{prox}_{\mathcal{R}^*}(w)_j = \begin{cases} \varepsilon & w_j > \varepsilon \\ w_j & |w_j| \leq \varepsilon \\ -\varepsilon & w_j < -\varepsilon \end{cases},$$

for all $j \in \{1, \ldots, s\}$. Then (24) reduces to

$$(\nabla \mathcal{R}_\Psi(u))_j = \gamma \left(A^\top v\right)_j \quad \text{with} \quad v_i := \begin{cases} \varepsilon & (Au+b)_i > \varepsilon \\ (Au+b)_i & |(Au+b)_i| \leq \varepsilon \\ -\varepsilon & (Au+b)_i < -\varepsilon \end{cases},$$

for $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, n\}$. If, in return, we choose $\mathcal{R}(u) = \chi_{C_3}(u)$, we recover the elementwise soft-shrinkage activation function $\operatorname{prox}_{\mathcal{R}^*}(x) = \operatorname{Softshrink}_\varepsilon(x)$ with

$$(\operatorname{Softshrink}_\varepsilon(x))_i := \begin{cases} x_i + \varepsilon & x_i < -\varepsilon \\ 0 & |x_i| \leq \varepsilon \\ x_i - \varepsilon & x_i > \varepsilon \end{cases},$$

in which case (24) reduces to $\nabla \mathcal{R}_\Psi(u) = \gamma A^\top \operatorname{Softshrink}_\varepsilon(Au+b)$. Finally, if we choose $\mathcal{R}$ such that $\mathcal{R}^*$ is the following

$$\mathcal{R}^*(w) = \begin{cases} w \tanh^{-1}(w) + \frac{1}{2}\left(\log(1-w^2) - w^2\right) & |w| < 1 \\ \infty & |w| \geq 1 \end{cases},$$

we get $\operatorname{prox}_{\mathcal{R}^*}(x) = \tanh(x)$ with

$$(\tanh(x))_i = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}}.$$

In this case, (24) reduces to $\nabla \mathcal{R}_\Psi(u) = \gamma A^\top \tanh(Au+b)$.

3.3 Inverse problems

We compare deep equilibrium and bilevel learning for the three different inverse problems of denoising, inpainting and deblurring, which we briefly describe in the following subsections.

3.3.1 Denoising

In the denoising task we assume measured data f is affected by additive noise δ only. This means that $K = I_n$, where $I_n \in \{0,1\}^{n \times n}$ is the identity matrix of size n, so equation (1) reads

$$u + \delta = f^\delta.$$

It is also clear that u and f^δ have the same dimension in this setting. The main goal is to remove the noise δ to retrieve u from f^δ.

3.3.2 Inpainting

In (image) inpainting we have a partial observation f of unknown, complete data u. For example, an image can have missing pixels when a user may want to remove an undesired object from said image. The task of filling missing pixels is called inpainting, and it can be modeled as an inverse problem [36]. The matrix K can be modeled by removing every row that corresponds to a missing pixel from an identity matrix I_n. If we want u and f^δ to have the same dimension (i.e., $m = n$), it suffices to substitute zero instead of one in each row that corresponds to a missing pixel.

3.3.3 Deblurring

In (image) deblurring the goal is to recover an unknown image u from a blurred and usually noisy image f^δ. This process can mathematically be described with the inversion of a convolution operator, which without regularization is highly ill-conditioned. If we want to apply convolutions to images, we can consider $U \in \mathbb{R}^{n_{\text{row}} \times n_{\text{col}}}$, where $n_{\text{row}} * n_{\text{col}} = n$, and $n_{\text{row}}, n_{\text{col}} \in \mathbb{N}$. We consider $u \in \mathbb{R}^n$ in our notation, which we obtain by turning U into a column vector consisting of all columns of U stacked in consecutive order.

A two-dimensional convolution operator K_Ω is then defined as $K_\Omega U = F^\delta_{\text{img}}$, with $[F^\delta_{\text{img}}]_{h,k} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} [\Omega]_{i,j} [U]_{h-i,k-j}$ for all h, k, where Ω is a fixed *convolution kernel*. The components of Ω are usually normalized to guarantee $\sum_i \sum_j [\Omega]_{i,j} = 1$.

3.4 Implementation

For the bilevel learning, we want to obtain the fixed point solution u^* defined in (15). We do this by initializing u_0 as the zero vector without loss of generality, and then we iteratively use (14). Note that any initial value for u_0 can be chosen because of convergence guarantees to the unique fixed point u^*.

The computation of the fixed point through (15) is computed until at least one of two stopping conditions is satisfied: either the relative tolerance $\|u^k - u^{k-1}\|/\|u^k\|$ is below a threshold, or the maximum number of iterations is reached. We choose 10^{-3} as threshold, and 500 as maximum number of iterations for the MNIST dataset because we experimentally discover that we can get a good trade-off between the quality of the reconstructed image and the computation time with this setting. For similar reasons, we choose 10^{-14} as threshold, and $1,000$ as maximum number of iterations for the CelebA dataset. To accelerate the convergence towards the fixed point solution, Anderson acceleration [37] can be implemented following the approach shown in [3]. However, in this paper we decided not to use it as it could lead to an ill-conditioned problem if the parameters are not carefully chosen.

For deep equilibrium regularizers, we solve (21), where $\mathcal{N}_\Psi$ works as a parametrized regularizer defined in (22). This means that the neural network we are using is composed of one affine input layer with activation function σ, and one linear operator as output layer with no bias nor activation function, scaled by a scalar γ which we consider as a hyperparameter chosen a-priori (i.e., it is not trained).

For bilevel learning models, we use the same setting with the additional constraint $C = A$ for the reasons explained in Section 3.2. To have a fair comparison, parameters in C are initialized to be equal to A for both the bilevel model and the deep equilibrium one, although during training only the bilevel model is constrained to satisfy $C = A$.

We want to remark that we choose $\nabla \mathcal{R}_\Psi$ to be the composition of one fully connected affine layer, an activation function, and a linear operator. This is done following Section 3.2 to guarantee a fair comparison between the performance of bilevel learning and DEQ models. More expressive architectures can be used to achieve better results, as shown in [2, 3, 23].

The chosen optimizer for training parameters Ψ is Adam [38] with initial learning rate 10^{-3}.

4 Numerical results

We compare bilevel optimization method and deep equilibrium model in our numerical experiments[1]. In particular, we want to experimentally verify whether one method works better than the other in terms of test error and robustness to tuning parameters. The model used for the bilevel optimization examples is defined in (14). For deep equilibrium models, we use (21)-(22).

We use the MNIST dataset [39], a dataset of digits that contains 70,000 grayscale images (60,000 for training, and 10,000 testing) of size 28×28, and CelebA [40], a dataset of 202,599 RGB face images of size 178×218. Given the larger amount of images in the CelebA dataset and their larger size with respect to MNIST, we randomly select 10 images for training and 10 for testing. For simplicity, we also decide to convert CelebA images to grayscale. We rescale pixel values to be within the range $[-1, 1]$. Those images are the unknown u we want to retrieve in (3).

We now describe the settings for the MNIST dataset, for which a broad analysis of hyperparameter choices has been performed. The selection of settings for the CelebA dataset is detailed in subsection 4.7. To generate the input f^δ, operator K (either denoising, inpainting, or deblurring described in Section 3.3) is applied to u. To avoid committing an inverse crime, instances δ of Gaussian random variables are added, where $\delta_i \sim \mathcal{N}(0, \alpha^2)$, and α is the noise level hyperparameter. We use $\alpha = 0.0, 0.05, 0.5$ for noise levels in training, and we keep the same value of α for the test dataset. We note that instances δ of Gaussian random variables are not generated once for each image; instead we generate new values of δ at each training epoch.

For the regularizers, the hyperparameters are set to the values $\tau = 0.01, 0.1, 0.9, 1.1, 2.1$, and $\gamma = 0.1, 0.5, 1.0$. We initialize A and C as square matrices with dimension 784×784. It follows that the bias b has dimension 784. This is not the only possible choice since it is sufficient that A and C have the same dimension (e.g., they can be fully-connected affine layers with rectangular matrices). We choose the activation function σ as ReLU, Softshrink, or identity, as discussed in Section 3.2. The value for the threshold of the Softshrink activation function is chosen as τ, although in principle they are different parameters and can be chosen differently.

We choose the mean-squared error (MSE) as the loss function to minimize the Euclidean distance between the original images $u^\dagger$ and the reconstructed images u^*.

The simulations are coded using Python (v3.10.7), and in particular the PyTorch (v1.12.1) library. All the simulations run on the GPU, which allows remarkable speed-up in terms of computation time compared to running the same experiments on the CPU. The code has been written and tested on a Windows 10 laptop, and the simulations were run on the High Performance Cluster system Apocrita [41].

[1] Our code is available at https://github.com/reacho/deep-equilibrium-vs-bilevel

4.1 Denoising

We start analyzing the denoising task. As it can be expected, our experiments suggest that the denoising task is the easiest among the three tasks considered. Indeed, we see from Fig.1 that the average loss of the trained model is lower in the denoising task, and the interquantile range is smaller too.

The visual comparison between bilevel optimization method and deep equilibrium model is shown in Fig.2. The original image (without noise) can be recovered by both methods, and there are no major differences between the reconstructions of the two methods. All the models that achieve a loss smaller than 0.5 achieve similar visual results.

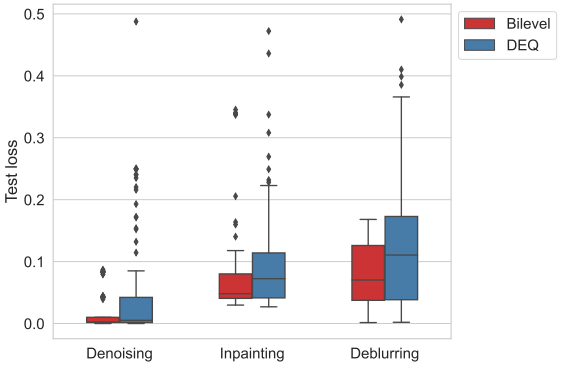


Figure 1: Comparison between bilevel optimization and deep equilibrium models for each of the three considered inverse problems, namely denoising, inpainting, and deblurring, over all the range of possible parameters. These boxplots consider the loss of the trained models evaluated on the test dataset. We removed all the configurations with a final loss larger than 0.5, a value we arbitrarily chose by looking for an empirical relation between the loss and the image quality.

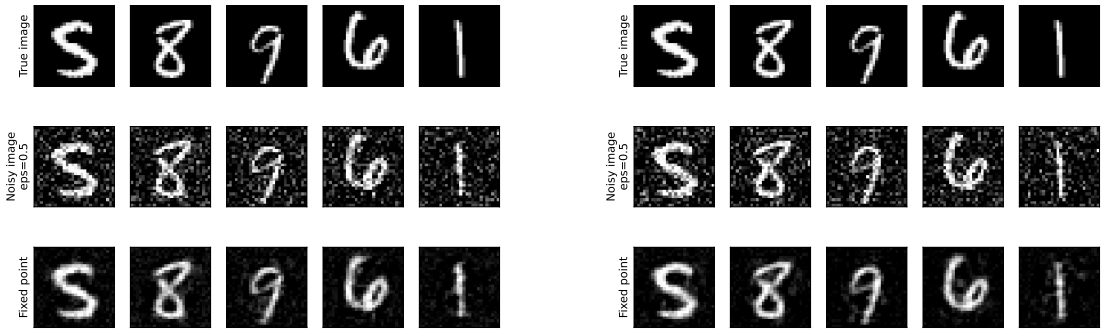


Figure 2: Denoising the MNIST dataset. Visual comparison between bilevel method (left) and deep equilibrium model (right), with parameters $\tau = 0.5$, $\gamma = 0.1$, and $\sigma =$ (ReLU). Images are taken from the test dataset. The first row shows the original images; the second row is the model input. The last row is the output of the trained models.

4.2 Inpainting

For inpainting, we choose to mask one third of the image rows, starting from the top and rounding up to the nearest integer. For MNIST images, this corresponds to masking 10 rows out of 28. Note that this task is harder than recovering the same number of pixels if they were randomly selected. In this latter scenario, the values of each missing pixel can be reasonably guessed by looking at the values of its neighbors, which is not true in our setting because most of the missing pixels are surrounded by other missing pixels. Only 117 hyperparameter-settings out of 225 worked for the bilevel model. Deep equilibrium models are more sensitive to the hyperparameter selection than bilevel methods for the inpainting task, since only 73 hyperparameters settings out of 225 lead to satisfactory results. We see from Fig.1 that the chosen bilevel methods seem to perform better than their deep equilibrium counterparts in terms of loss minimization.

Visual results are shown in Fig.3. Interestingly, both methods achieve good images reconstructions.

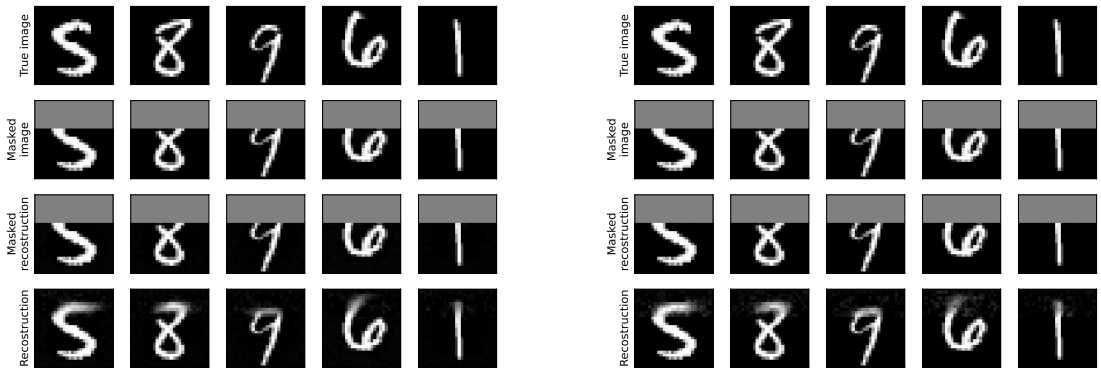


Figure 3: Inpainting MNIST. Comparison between bilevel method (left) and deep equilibrium model (right), with parameters $\tau = 0.5$, $\gamma = 1.0$, and $\sigma =$ (Softshrink). Images are taken from the test dataset. The first row shows the original image, the second row is the masked image, i.e., the input of the algorithm. The fourth row is the output of the trained models. Finally, the third row shows what happens when we apply the inpainting operator on the output. The fourth row is the output of the trained deep equilibrium optimization problem. Ideally, the difference between the second and third row should be small.

4.3 Deblurring

The last task we consider is deblurring. We model the convolution operator to mimic a diagonal motion blur, for which we choose the convolution kernel Ω as $\Omega = \frac{1}{5} I_5$, where I_5 denotes the 5×5 identity matrix. This task has the largest variability in terms of loss results for trained models, as shown in Fig.1. We also see that bilevel methods achieve a smaller average loss in comparison to deep equilibrium models, with also a smaller interquantile range. The quality of the retrieved images is similar for the two methods as shown in Fig.4, even though the average loss seems to suggest a different result (this is because the mean-squared error is not always a good indicator of *similarity* between a pair of images).

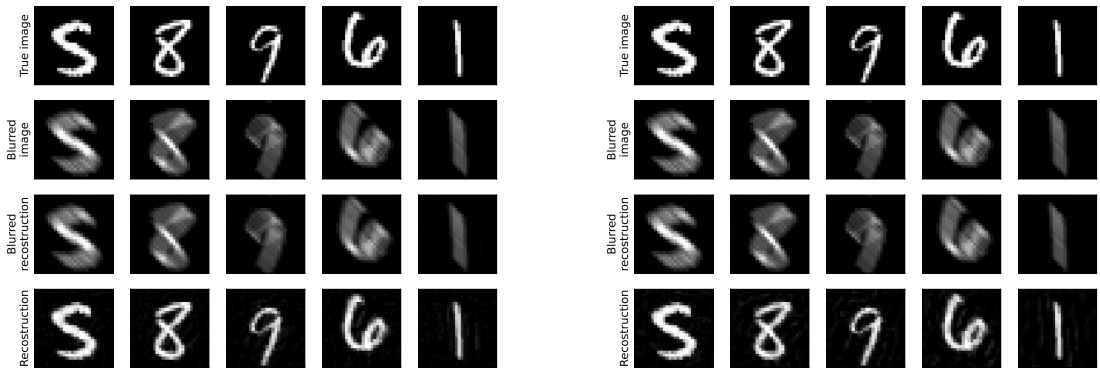


Figure 4: Deblurring MNIST. Comparison between bilevel method (left) and deep equilibrium model (right), with parameters $\tau = 0.5$, $\gamma = 0.5$, and $\sigma =$ (Softshrink). Images are taken from the test dataset. The first row shows the original images; the second row is the model input. The last row is the output of the trained models. The third row shows the model output after we apply the convolution kernel to it. Ideally, the difference between the second and the third rows should be small.

4.4 Sensitivity against noise level

The number of parameters in deep equilibrium models is larger than the one for bilevel methods in the framework we considered. Therefore, we want to assess whether this has an impact on the loss for different noise levels. To do so, we visualize the loss for both training and test datasets. Results are shown in Fig.5. Both methods seem to have the same behavior for increasing noise level values. Training takes longer for larger noise levels, and the loss becomes larger and larger as expected.

4.5 Sensitivity against hyperparameter selection

Deep equilibrium models have twice as many parameters as bilevel learning methods in our setting. For our computations, we capped the running time to at most one hour, which is why we need to evaluate the impact of this choice on the results. To do so, we consider all the simulations, keeping only those with a final test loss smaller than 0.5, and collect the number of epochs in bins in a histogram. The resulting histograms are shown in Fig.6. The shapes of bilevel and deep equilibirum histograms are similar, suggesting that capping the running time to one hour for the simulations does not have an impact on the quality of the results. The different scale of the y axes for the two plots reflects that more simulations for the deep equilibrium models have exceded the loss threshold of 0.5 in comparison to the bilevel methods. This further supports that the chosen bilevel methods are less sensitive to hyperparameter selection than their deep equilibrium counterparts.

4.6 Naïvely learning the fixed-point

In this section we show the results we get if we choose to train the model by naïvely learning the fixed-points, as explained in Section 2.5. Here we train the parameters by minimizing (20). To easily impose convergence to the fixed point, we perform this test with the bilevel optimization model. In order to compare the reconstruction with the one we show in Fig.3,

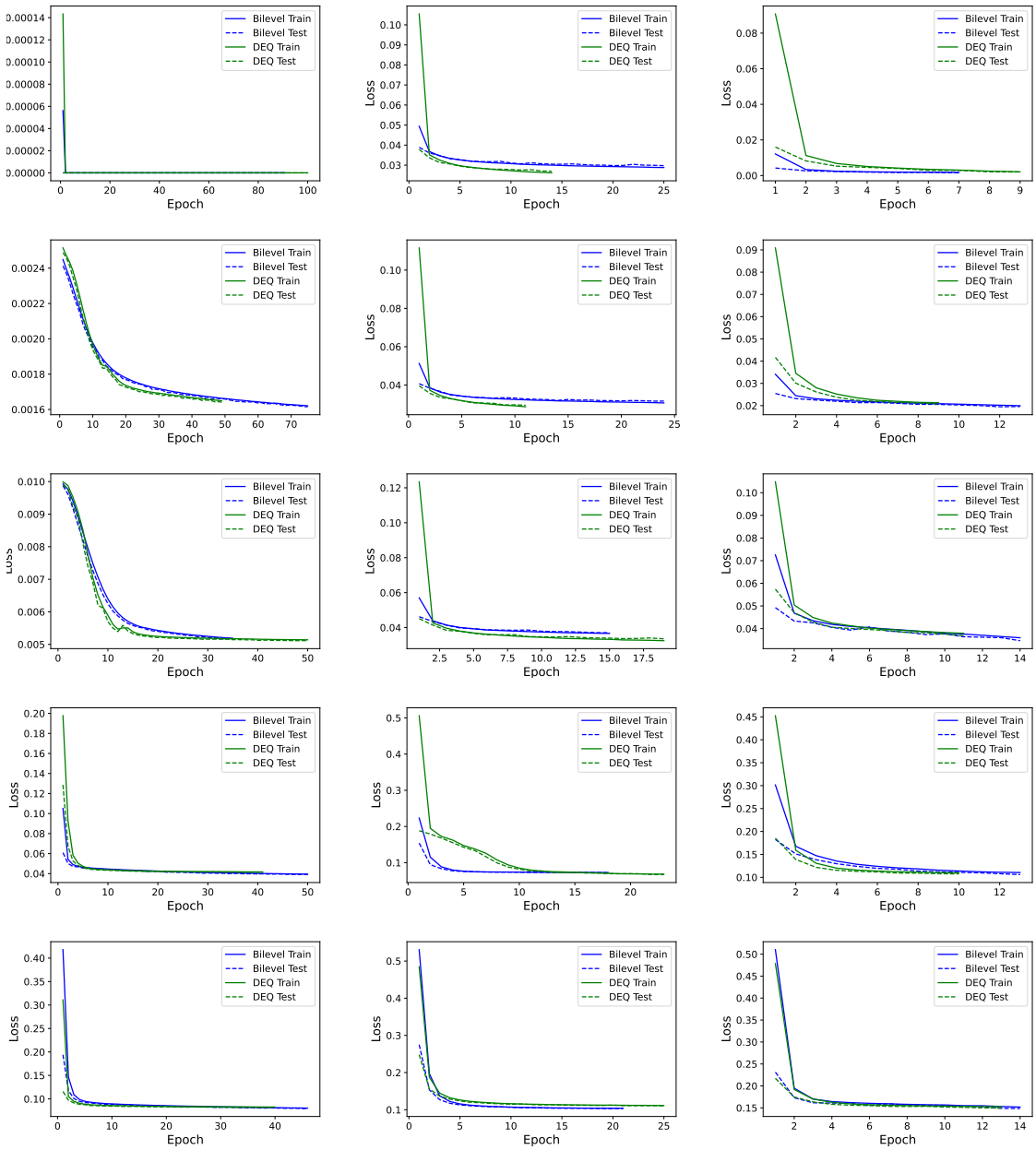


Figure 5: Comparison of the loss error for the test dataset evaluated after each training epoch, for increasing values of noise levels in training (noise levels from top to bottom row: $0, 0.05, 0.1, 0.5, 1$). Simulations are grouped by the tasks, namely denoising, inpainting, and deblurring (left, center, right columns). Each plot shows the simulation with the configurations that achieve the lowest final test loss.

we choose the same settings (namely, $\tau = 0.5$, $\gamma = 1.0$, and $\sigma = \text{Softshrink}$).

After training the parameters using (20), we check whether the trained model is able to fill the missing pixels from a masked image. To do so, we initialize u_0 in the same way explained at the beginning of Section 3.4 and we run 100 iterations of (15). The results are shown in Fig.7, where it is clear that missing regions are not filled. We again emphasize that the model

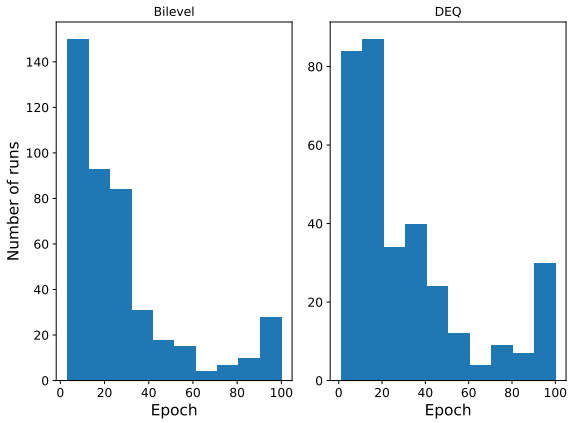


Figure 6: These histograms show how many simulations were finished within an hour as a function of the number of epochs. Each simulation is a different configuration of hyperparameters. We consider only those runs where the loss on the test dataset is smaller than 0.5.

and the hyperparameters are the same ones used in Fig.3; the reconstruction is satisfactory in the latter. The only difference between the two models is the way in which the parameters have been learned. This empirically supports the claims made in Section 2.5.

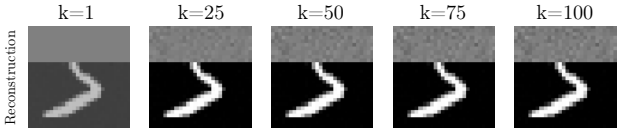


Figure 7: Reconstruction for the inpainting task for a bilevel optimization model whose parameters have been trained by minimizing the error of the reconstruction u^* w.r.t. the true image $u^\dagger$ (naïve approach). We show how the reconstruction $\{u^k\}$ changes for different values of the iteration k. As we can see, the model trained with the naïve approach is not able to inpaint the masked area.

4.7 Comparison on a higher-resolution dataset

We now compare how bilevel learning and DEQ model perform on CelebA, a higher-resolution dataset. For simplicity, the images are converted to grayscale, and pixel values are normalized in the range $[-1, 1]$. Because of the dimension of the images, using fully-connected dense matrices for A and $C^\top$ is not a viable option, as it would lead to memory issues. Instead, we use 2D convolutional layers, allowing us to have fewer parameters. We choose to omit the inpainting task for this dataset, as it necessitates non-local information that convolutional layers cannot provide, in contrast to fully-connected dense layers.

4.7.1 Denoising

Motivated by the Rudin Osher Fatemi (ROF) model [42], we modify (21) and (22) to

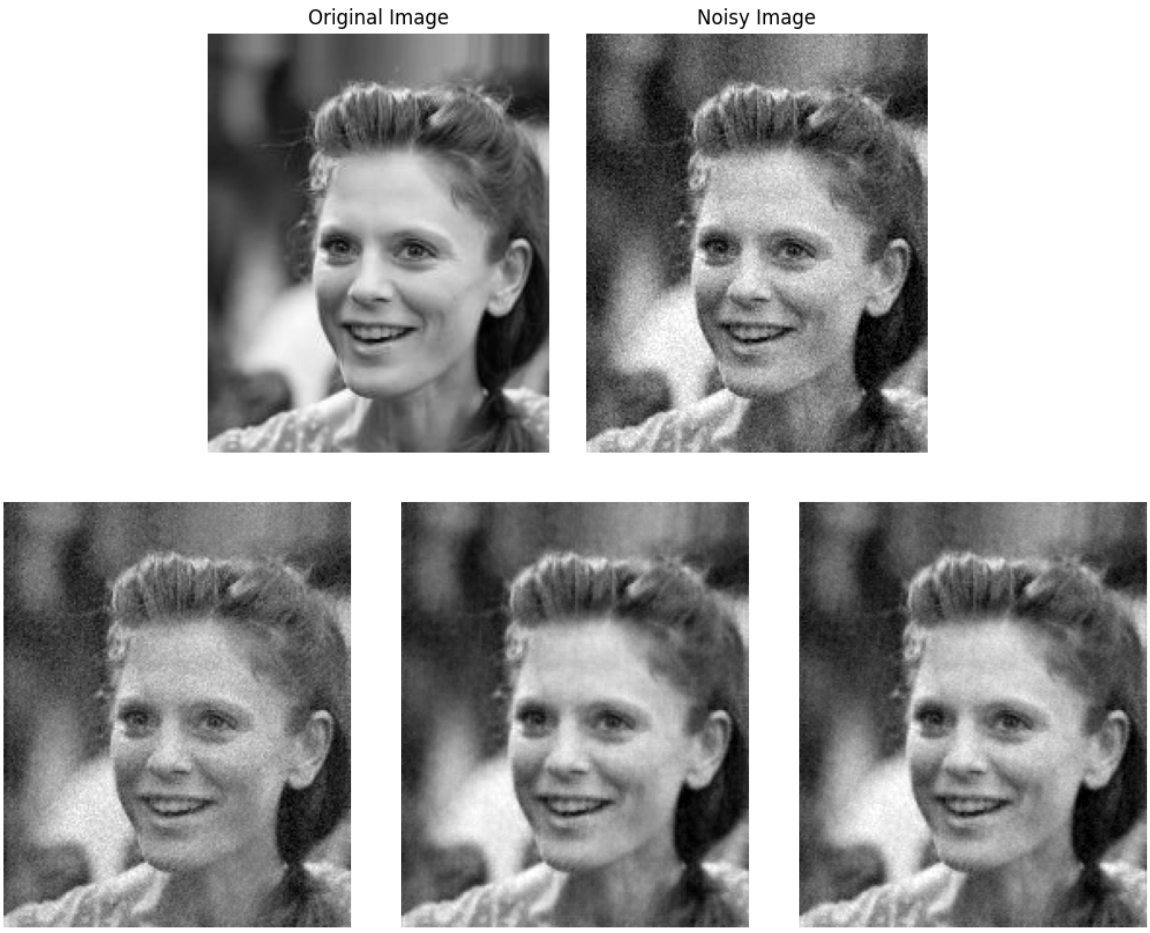
$$u^{k+1} = u^k - \tau \left(\lambda K^\top \left(K u^k - f^\delta \right) + \gamma C^\top \sigma(\xi A u^k) \right)$$

with $\sigma(\cdot) = \tanh(\cdot)$, and $\xi \in \mathbb{R}$ 'large enough' to approximate the *sign* activation function. Choosing $\xi = 100$, $\gamma = 1$, the value of λ that empirically minimizes the MSE for noise with standard deviation 0.1 is $\lambda = 18.156$. Please note that for the anisotropic ROF model, $\mathcal{R}(u)$ reads $\mathcal{R}(u) = \|\nabla u\|_1$, where $\|\cdot\|_1$ denotes the one-norm and ∇ a discretization of the gradient operator. Hence $\nabla^\top sign(\nabla u)$ is a subgradient of this function, and replacing ∇ and $\nabla^\top$ with operators A and C and *sign* with the hyperbolic tangent yields a trainable model in the vein of ROF denoising.

We consider different initializations for the denoising task, with either 2 or 30 output channels and kernels of size 11×11 or 3×3 kernel, respectively. This is done to test two extreme scenarios: either having a low number of output channels with relatively large kernels, compared against many output channels with small kernels. The number of input channels is dataset-dependent, and for grayscale images, it is one. The kernel weights of A and C are initialized with the same values for both Deep Equilibrium models and bilevel learning, whereas in the latter we add the constraint that $C = A$ through all the training.

We use Adam as the optimizer, together with a scheduler which makes the learning rate decrease linearly over the epochs, starting in the range $\left[3.2 \cdot 10^{-3}, 10^{-2}\right]$ and ending with a value which is 100 times smaller. This is done to help escaping local minima in the first epochs, and to converge faster towards the end of the training. All simulations are run over 500 epochs. For the computation of the fixed point we use a naïve forward iteration scheme that we stop when either $1,000$ iterations have been computed, or a relative norm difference between iterations below 10^{-14} has been achieved. We decide to perform spectral normalization for A and C, i.e., we divide the operators by their norms which are estimated using a power iteration scheme. An alternative to this approach is to avoid computing the spectral normalization. In the latter case, (21) may not converge unless an appropriate value of τ is chosen after every optimization step. To guarantee the convergence to the fixed point, we check the gradient values before performing the optimization step, and we decrease τ by an arbitrary factor of 10 if at least one component of the gradient becomes either ∞, or NaN; at the same time, we also increase the maximum number of iterations by the same factor to compute the fixed point (21).

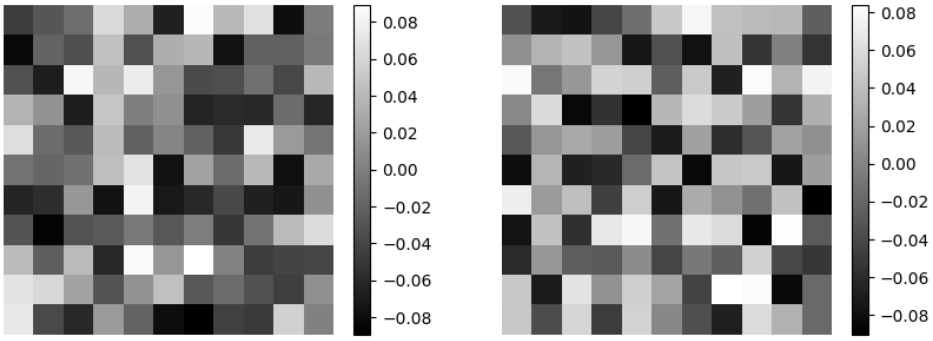
The reconstruction of the images from noisy samples is shown in Fig.8. The noise can be removed using either bilevel learning or DEQ models, with similar results in terms of MSE. Similarly, the quality of the reconstruction does not seem affected by the number of output channels and the size of the kernels, as long as the model is expressive enough. Interestingly, the kernels do not change too much even if the loss decreases. The major change is in the overall norm, whereas the relative values between the pairs of pixels remain similar (see Figs.9-10 for a comparison of how much the kernels change in bilevel learning against DEQ methods on 11×11 kernels, and Figs.11-12 for 3×3 kernels). This is true for both the TV-like initialized kernels, where we could have expected to see the pattern change from the second-order to the first-order initialization (or vice-versa), and for the randomly initialized. Although this result seems counter-intuitive as it would be reasonable to expect to see some patterns appear in the kernels, a possible explanation is that the patterns are hidden in the kernels and are just not easily interpretable. Furthermore, it seems that any couple of randomly initialized kernels can perform denoising, provided that the weights are slightly changed.



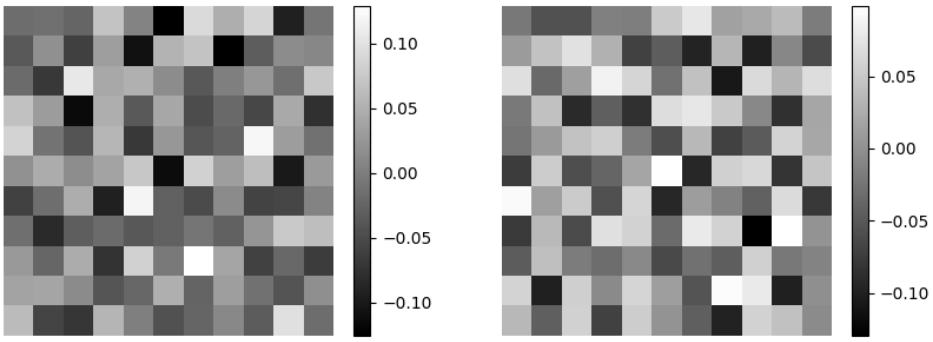
(a) Reconstructed images, two output channels, kernels size 11×11

(b) Reconstructed images, thirty output channels, kernels size 3×3

Figure 8: Denoising CelebA; sample from the test dataset. The first row contains the original image u and the noisy image f^δ. From left to right in the second and third rows: reconstructed image with random initializations of the kernels (left), with parameters learned using bilevel learning (center), and parameters learned using the DEQ model (right).

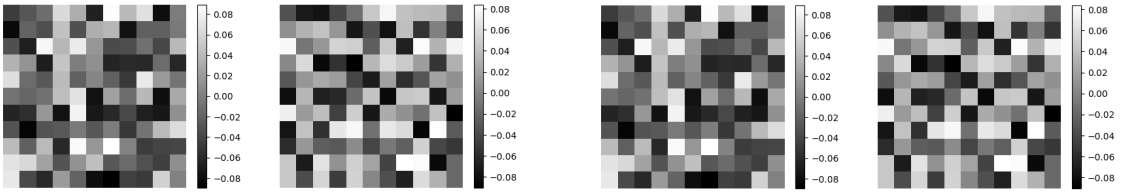


(a) 11×11 kernels before training.

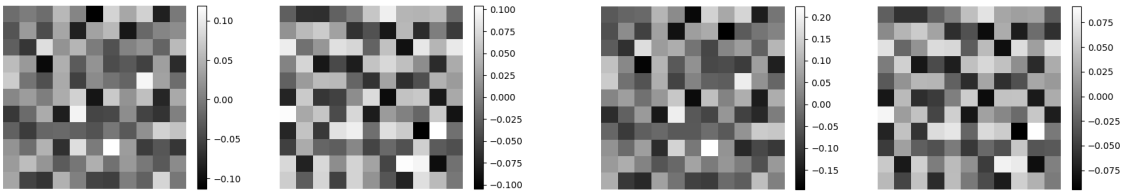


(b) 11×11 kernels after training.

Figure 9: Comparison of 11×11 kernels of A, shown before and after training on the denoising task using the bilevel learning model. The weights of $C^\top$ are not shown here since $C^\top = A^\top$ for bilevel learning models.

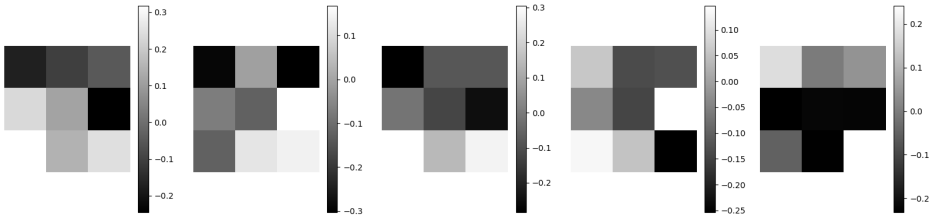


(a) 11×11 kernels before training.

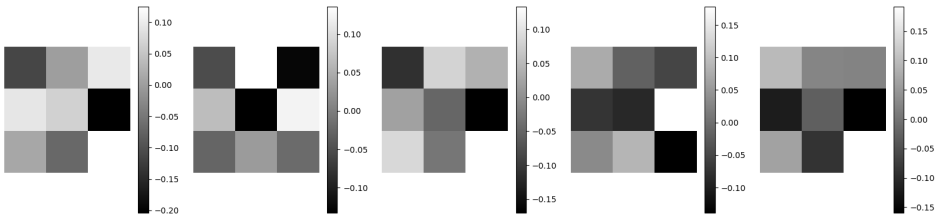


(b) 11×11 kernels after training.

Figure 10: Comparison of 11×11 kernels of A (first two columns on the left) and of $C^\top$ (two columns on the right), shown before and after training on the denoising task using the DEQ model. Note that, in the first row, the pairs of kernels in the first-third columns and the second-fourth are the same; this is because we initialize the kernels so that $C^\top = A^\top$ before training. After the training, they are different (second row).

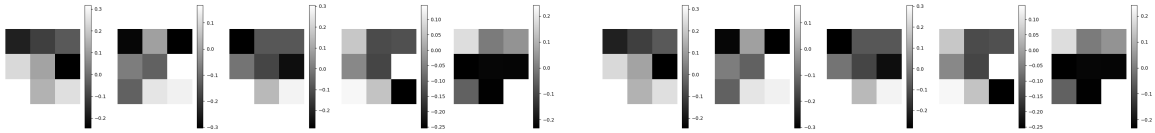


(a) 3×3 kernels before training.

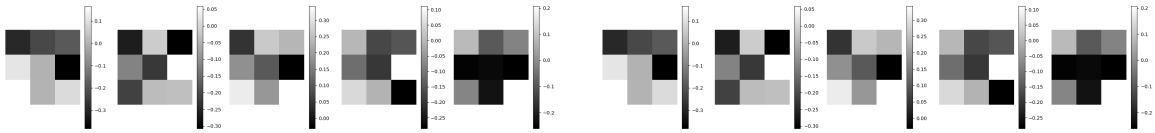


(b) 3×3 kernels after training.

Figure 11: Comparison of five (out of thirty) 3×3 kernels of A, shown before and after training on the denoising task using the bilevel learning model. The weights of $C^\top$ are not shown here since $C^\top = A^\top$ for bilevel learning models.



(a) 3×3 kernels before training.



(b) 3×3 kernels after training.

Figure 12: Comparison of 3×3 kernels of A (first five columns on the left) and of $C^\top$ (five columns on the right), shown before and after training on the denoising task using the DEQ model. Note that, in the first row, kernels are pairwise equal; this is because we initialize the kernels so that $C^\top = A^\top$ before training. After the training, they are different (second row).

4.7.2 Deblurring

The second inverse problem we consider for the CelebA dataset is deblurring. We perform a warm-start, loading the optimal kernels' weights learned in the denoising task. The reconstructed images are shown in Fig.13. Both bilevel learning and DEQ models achieve a similar quality in the reconstructed images. Considerations similar to the ones drawn in the denoising task can be done on the learned kernels for the deblurring task (Figs.14-15 show a comparison of how much the kernels change in bilevel learning against DEQ methods on 11×11 kernels, and Figs.16-17 for 3×3 kernels).

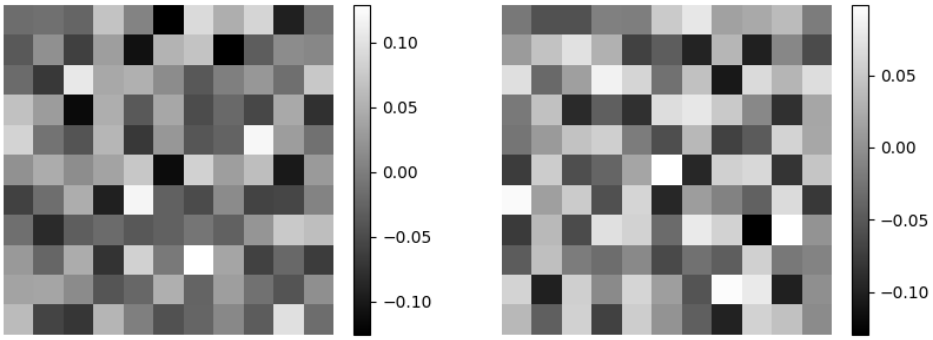
5 Conclusions & Outlook

We have framed bilevel learning methods as deep equilibrium methods and have compared several models of each class of similar complexity. From those numerical results, we have observed that the two methods behave similarly in terms of average loss when they are used as regularizers. We can even argue that regularizers trained by bilevel learning were performing slightly better than their deep equilibrium counterparts, which can have different reasons. The average loss of the trained models with bilevel learning is smaller, with a lower interquantile range than deep equilibrium models. The results also suggest bilevel learning is less sensitive to the choice of hyperparameters than deep equilibrium method, making it more robust in terms of hyperparameters selection, and easier to train. This is in contrast to the a-priori assumption that deep equilibrium models might perform better because they are more general than their bilevel learning counterparts. The number of parameters in the deep equilibrium models we considered is around twice than the ones of the bilevel learning models. The experiments suggest that the extra constraints imposed by requiring the regularizing network to be the gradient of a regularizer does not hinder the performance, and may even enhance it. We emphasize that these observations are limited to the gradient descent deep equilibrium architecture. Further comparisons with other deep equilibrium architectures are subject to future work. It would also be interesting to study deep equilibrium models mathematically in

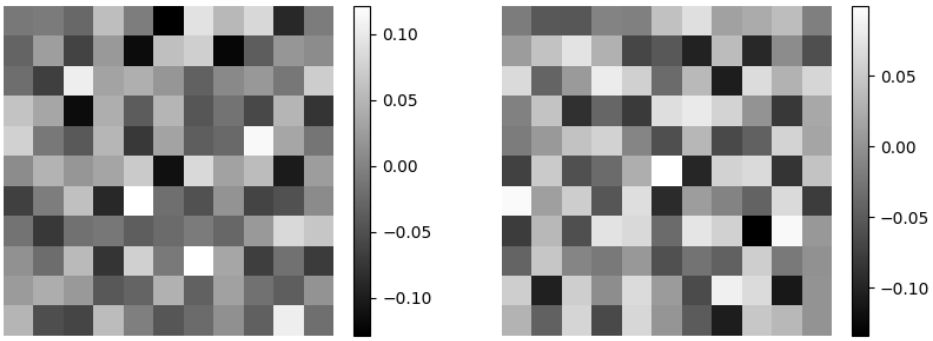
(a) Reconstructed images, two output channels, kernels size 11×11

(b) Reconstructed images, thirty output channels, kernels size 3×3

Figure 13: Deblurring CelebA, a sample from the test dataset. The first row contains the original image u and the noisy blurred image f^δ. From left to right in the second and third rows: reconstructed image with the optimal kernels found for the denoising task in the bilevel scenario (left), with parameters learned using bilevel learning (center-left), with parameters learned using DEQ model (center-right), and optimal kernels found for the denoising task in the DEQ scenario.

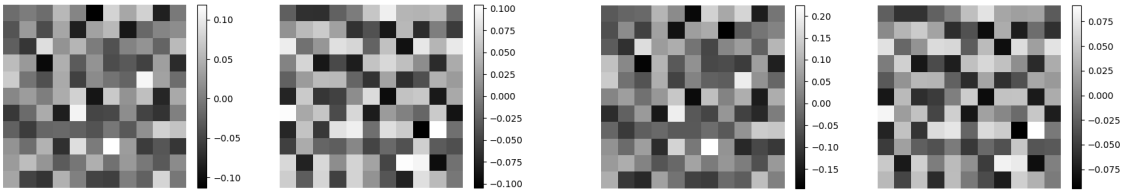


(a) 11×11 kernels before training.

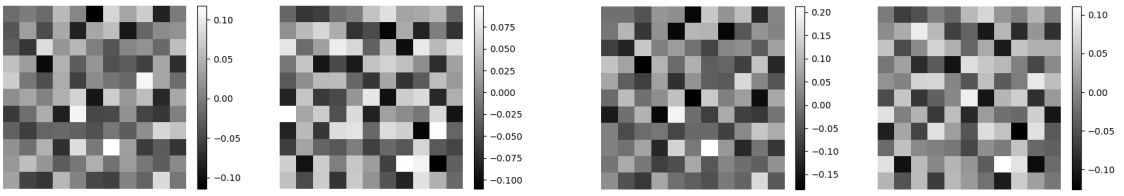


(b) 11×11 kernels after training.

Figure 14: Comparison of 11×11 kernels of A, shown before and after training on the deblurring task using the bilevel learning model. The weights of $C^\top$ are not shown here since $C^\top = A^\top$ for bilevel learning models.

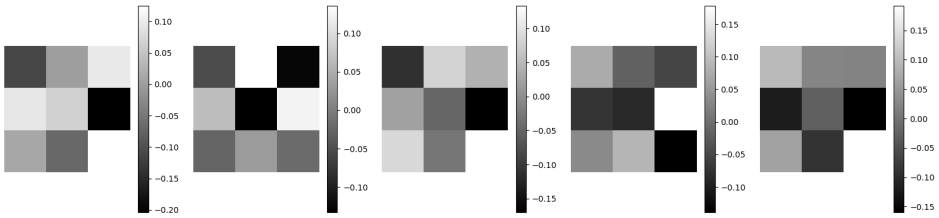


(a) 11×11 kernels before training.

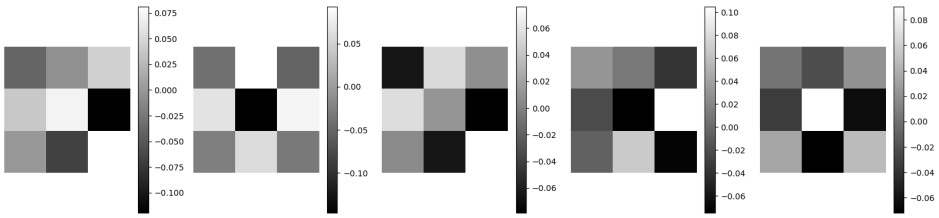


(b) 11×11 kernels after training.

Figure 15: Comparison of 11×11 kernels of A (first two columns on the left) and of $C^\top$ (two columns on the right), shown before and after training on the deblurring task using the DEQ model. Note that, in the first row, the pairs of kernels in the first-third columns and second-fourth are the same; this is because we initialize the kernels so that $C^\top = A^\top$ before training. After the training, they are different (second row).

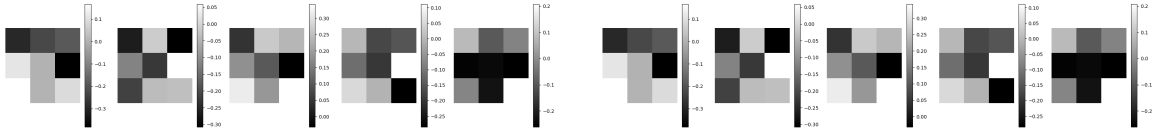


(a) 3×3 kernels before training.

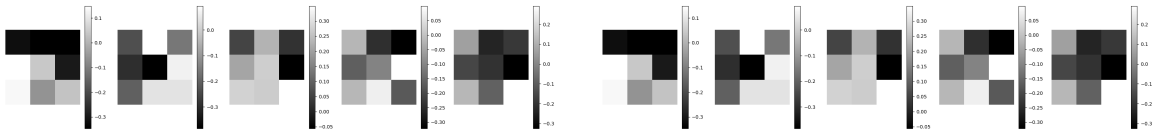


(b) 3×3 kernels after training.

Figure 16: Comparison of five (out of thirty) 3×3 kernels of A, shown before and after training on the deblurring task using the bilevel learning model. The weights of $C^\top$ are not shown here since $C^\top = A^\top$ for bilevel learning models.



(a) 3×3 kernels before training.



(b) 3×3 kernels after training.

Figure 17: Comparison of 3×3 kernels of A (first five columns on the left) and of $C^\top$ (five columns on the right), shown before and after training on the deblurring task using the DEQ model. Note that, in the first row, kernels are pairwise equal; this is because we initialize the kernels so that $C^\top = A^\top$ before training.

greater detail, and to identify conditions that can lead to theoretical guarantees for them.

Acknowledgements

The authors would like to thank the Isaac Newton Institute for Mathematical Sciences, Cambridge, for support and hospitality during the programme *Mathematics of Deep Learning* where work on this paper was undertaken. This work was supported by EPSRC grant no EP/R014604/1. This research utilized Queen Mary's Apocrita and Andrena HPC facilities, supported by QMUL Research-IT http://doi.org/10.5281/zenodo.438045. DR acknowledges support from EPSRC grant EP/R513106/1. MB acknowledges support from the Alan Turing Institute. MJE acknowledges support from the EPSRC (EP/S026045/1, EP/T026693/1, EP/V026259/1) and the Leverhulme Trust (ECF-2019-478).

References

[1] Stefan Roth and Michael J Black. "Fields of experts". In: *International Journal of Computer Vision* 82.2 (2009), pp. 205–229.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "Deep Equilibrium Models". In: *Advances in Neural Information Processing Systems 32*. 2019.

[3] Davis Gilton, Greg Ongie, and Rebecca Willett. "Deep Equilibrium Architectures for Inverse Problems in Imaging". In: *IEEE Transactions on Computational Imaging* (2021).

[4] Frank Natterer and Frank Wübbeling. *Mathematical methods in image reconstruction*. SIAM, 2001.

[5] Otmar Scherzer et al. *Variational methods in imaging*. Springer, 2009.

[6] A.N. Tikhonov, A. Goncharsky, and M. Bloch. "Ill-posed problems in the natural sciences". In: *Mir* (1987).

[7] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*. Vol. 375. Springer Science & Business Media, 1996.

[8] Martin Benning and Martin Burger. "Modern regularization methods for inverse problems". In: *Acta Numerica* 27 (2018), pp. 1–111.

[9] Juan Carlos De Los Reyes and Carola-Bibiane Schönlieb. "Image Denoising: Learning the Noise Model via Nonsmooth PDE-Constrained Optimization". In: *Inverse Problems and Imaging* 7 (2013), pp. 1183–1214.

[10] Karl Kunisch and Thomas Pock. "A Bilevel Optimization Approach for Parameter Learning in Variational Models". In: *SIAM Journal on Imaging Sciences* 6 (2 2013), pp. 938–983.

[11] Peter Ochs et al. "Bilevel Optimization with Nonsmooth Lower Level Problems". In: *SSVM*. Vol. 9087. 2015, pp. 654–665.

[12] J. C. De los Reyes, Carola-Bibiane Schönlieb, and T. Valkonen. "Bilevel Parameter Learning for Higher-Order Total Variation Regularisation Models". In: *Journal of Mathematical Imaging and Vision* 57 (1 2017), pp. 1–25.

[13] Matthias J. Ehrhardt and Lindon Roberts. "Inexact Derivative-Free Optimization for Bilevel Learning". In: *Journal of Mathematical Imaging and Vision* 63 (5 2021), pp. 580–600.

[14] Caroline Crockett and Jeffrey A. Fessler. *Bilevel Methods for Image Reconstruction*. 2021. URL: http://arxiv.org/abs/2109.09610.

[15] Karol Gregor and Yann LeCun. "Learning Fast Approximations of Sparse Coding". In: *27th International Conference on Machine Learning*. 2010, pp. 399–406.

[16] Kyong Hwan Jin et al. "Deep Convolutional Neural Network for Inverse Problems in Imaging". In: *IEEE Transactions on Image Processing* 26 (9 Sept. 2017), pp. 4509–4522. ISSN: 10577149.

[17] Bo Zhu et al. "Image Reconstruction by Domain-Transform Manifold Learning". In: *Nature* 555 (7697 2018), pp. 487–492.

[18] Jonas Adler and Ozan Öktem. "Learned Primal-Dual Reconstruction". In: *IEEE Transactions on Medical Imaging* 37 (6 2018), pp. 1322–1332.

[19] Jonas Adler and Ozan Öktem. "Solving ill-posed inverse problems using iterative deep neural networks". In: *Inverse Problems* 33 (2017), p. 124007.

[20] Erich Kobler et al. "Variational networks: Connecting variational methods and deep learning". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10496 LNCS (2017), pp. 281–293.

[21] Guang Yang et al. "DAGAN: Deep De-Aliasing Generative Adversarial Networks for Fast Compressed Sensing MRI Reconstruction". In: *IEEE Transactions on Medical Imaging* 37 (6 2018), pp. 1310–1321.

[22] Erich Kobler et al. "Total Deep Variation: A Stable Regularizer for Inverse Problems". In: *accepted by IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[23] Howard Heaton et al. "Feasibility-based fixed point networks". In: *Fixed Point Theory and Algorithms for Sciences and Engineering* 2021.1 (2021), pp. 1–19.

[24] Andrei Nikolaevich Tikhonov. "On the solution of ill-posed problems and the method of regularization". In: *Doklady Akademii Nauk*. Vol. 151. 3. Russian Academy of Sciences. 1963, pp. 501–504.

[25] Yunjin Chen, Rene Ranftl, and Thomas Pock. "Insights into analysis operator learning: From patch-based sparse models to higher order MRFs". In: *IEEE Transactions on Image Processing* 23.3 (2014), pp. 1060–1072.

[26] Patrick L Combettes and Jean-Christophe Pesquet. "Fixed point strategies in data science". In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 3878–3905.

[27] Juan Carlos De los Reyes, C-B Schönlieb, and Tuomo Valkonen. "The structure of optimal parameters for image restoration problems". In: *Journal of Mathematical Analysis and Applications* 434.1 (2016), pp. 464–500.

[28] Eberhard Zeidler. *Applied functional analysis: applications to mathematical physics*. Vol. 108. Springer Science & Business Media, 2012.

[29] Boris S Mordukhovich. *Variational analysis and generalized differentiation II: Applications*. Vol. 331. Springer, 2006.

[30] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*. Vol. 408. Springer, 2011.

[31] Jean Jacques Moreau. "Fonctions convexes duales et points proximaux dans un espace hilbertien". In: *Comptes rendus hebdomadaires des séances de l'Académie des sciences* 255 (1962), pp. 2897–2899.

[32] Kösaku Yosida. *Functional analysis*. Springer, 1964.

[33] Jean-Jacques Moreau. "Proximité et dualité dans un espace hilbertien". In: *Bulletin de la Société mathématique de France* 93 (1965), pp. 273–299.

[34] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[35] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *ICML*. 2010.

[36] Carola-Bibiane Schönlieb. *Partial differential equation methods for image inpainting*. Vol. 29. Cambridge University Press, 2015.

[37] Homer F. Walker and Peng Ni. "Anderson Acceleration for Fixed-Point Iterations". In: *SIAM Journal on Numerical Analysis* 49.4 (2011), pp. 1715–1735.

[38] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[39] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *MNIST handwritten digit database*. 1998. URL: http://yann.lecun.com/exdb/mnist/.

[40] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[41] Thomas King, Simon Butcher, and Lukasz Zalewski. *Apocrita - High Performance Computing Cluster for Queen Mary University of London*. Mar. 2017. URL: https://doi.org/10.5281/zenodo.438045.

[42] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.